2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)

A Protocol Independent Approach in Network Covert Channel Detection

Md. Ahsan Ayub*, Steven Smith†, Ambareen Siraj‡
Department of Computer Science
Tennessee Technological University
Cookeville, USA
{mayub42*, smsmith23†}@students.tntech.edu
asiraj@tntech.edu‡

***Abstract*—Network covert channels are used in various cyber-attacks, including disclosure of sensitive information and enabling stealth tunnels for botnet commands. With time and technology, covert channels are becoming more prevalent, complex, and difficult to detect. The current methods for detection are protocol and pattern specific. This requires the investment of significant time and resources into application of various techniques to catch the different types of covert channels. This paper reviews several patterns of network storage covert channels, describes generation of network traffic dataset with covert channels, and proposes a generic, protocol-independent approach for the detection of network storage covert channels using a supervised machine learning technique. The implementation of the proposed generic detection model can lead to a reduction of necessary techniques to prevent covert channel communication in network traffic. The datasets we have generated for experimentation represent storage covert channels in the IP, TCP, and DNS protocols and are available upon request for future research in this area.**

***Index Terms*—Decision Tree, DNS, IPv4, K-Nearest Neighbors, Logistic Regression, Network Covert Channel, Support Vector Machine (SVM), TCP**

I. Introduction

Network covert channels are a hidden form of communication that use existing protocols in unintended ways for unauthorized information transfer. There are two types of covert channels: storage covert channels (which use the direct or indirect writing of a storage location by one process and the direct or indirect reading of it by another) and timing covert channels (which use the timing of packets/processes to represent the communication being sent over the channel). Researchers have been studying these channels since 1973 [7].

Covert channels can be used to secretly leak sensitive data and hide military as well as secret service communication [10]. Wendzel *et al.* [18] created a taxonomy that categorizes patterns for such techniques. Our focus is on network storage covert channels without consideration of the payload of the packets. These channels can enable indiscernible security breaches for a network. Examples of storage covert channel techniques include the embedding of hidden data using random values, encoding data into the least significant bit (LSB) of a field, and insertion of data in unused portions of header fields. There has been a considerable amount of work done to illustrate effective methods for the detection of covert channels in a specific network protocol; however, little work has been done to propose a protocol independent approach for network covert channel detection [11], [12], [16].

Many techniques have been proposed and implemented to detect the actions of covert channels using machine learning techniques. Wendzel *et al.* [18] claim that covert channel patterns are generic across protocols and can therefore be identified for network, transport, and application layers if certain parameters match. However, very few general detection techniques exist. In addition, there are studies to detect the presence of network covert channels using Neural Network [17] and Support Vector Machines (SVM) [15]. Although these machine learning techniques can be used in layers beyond the transport layer to improve covert channel detection, at network level, these studies are limited to the TCP sequence number field and/or the IP identification field. They do not consider other possibilities of intentional hidden communication using various other network protocol level parameters.

Traditionally, detection across multiple protocols requires the simultaneous use of multiple techniques, leading to high overhead on network systems. In this study, we seek to identify a common approach for detection of covert channels across the various protocols of a network. This will eliminate the need for the detection mechanisms to be tailored for a specific protocol. In addition, we also describe the generation of an experimental network traffic dataset containing covert channels. Our generated dataset (available upon request) can potentially be used to assist the research community with improving techniques for the protection of critical systems and information against network covert channel attacks. We implemented supervised machine learning techniques that demonstrated excellent results for detection. Our collective research findings can be used to improve methods for the protection of critical systems against covert channel attacks.

This paper is organized as follows: Section 2 reviews related works in the field of network covert channel identification and detection. Section 3 explains our protocol independent approach to network covert channel detection and the methods used to generate the experimental network covert channel datasets. Section 4 presents the results of the research followed by discussion in Section 5. Section 6 summarizes the paper, its contributions, and future work to improve upon this research.

978-1-7281-1664-8/19/$31.00 ©2019 IEEE
DOI 10.1109/CSE/EUC.2019.00040

II. Related Work

Covert channels were first defined in research in the study "A note on the Confinement Problem" by B.W. Lampson in 1973 [8]. He defined them as channels not intended for information transfer. Over the last few decades, researchers have found success with the application of machine learning techniques for covert channel detection. Sohn *et al.* [15] achieved high detection rates in identifying network covert channels with Support Vector Machines (SVM). The use of Neural Networks for the detection of TCP Sequence Number covert channels was also proposed [17]. Their example created a trained model based on a normalized profile for detection of anomalous TCP Sequence numbers. The model detected covert channels by identifying packets containing TCP Sequence numbers that did not match the model's profile for that operating system. They were able to achieve a high level of precision of approximately 98% accuracy for most operating system environments in their experiments.

Zander *et al.* [19] sought to gather the currently known network covert channel techniques into one collection as well as methods to detect and eliminate them as reported in their paper "A Survey of Covert Channels and Countermeasures in Computer Network Protocols". The channels were categorized into different types, *e.g.*, header extensions and packet rate/timing channels. Many methods of detection were examined including information flow analysis, non-interference analysis, and covert flow trees. Countermeasures such as limiting bandwidth, improving network security, and traffic normalization were explored along with the use of classifiers for detection.

Wendzel *et al.* [18] organized patterns of network covert channels into hierarchical categories derived from 109 different covert channel techniques from several different studies. They found that the majority of covert channel techniques (69.7%) were based on four distinct patterns that modified attributes within a network packet. Our work focuses on the three network storage covert channel patterns that best represent the majority of known storage covert channel techniques.

Many researchers generated their own network covert channel datasets for experimentation purposes; however, they are not available for public use. This fact as well as the need to focus on specific covert channel patterns led us to generate our own dataset that serves our evaluation needs. This dataset will be available upon request for use by the research community.

III. Methodology

An experimental setting has been created by generating covert traffic data for the network layer (IPv4), transport layer (TCP), and application layer (DNS). The dataset includes covert channel traffic along with regular or benign traffic. To combat the complexity of network packets and extract the embedded covert information, we perform feature engineering in each aforementioned layer. This emphasizes the most important features for detection by supervised classification. Further details are provided in the following sections.

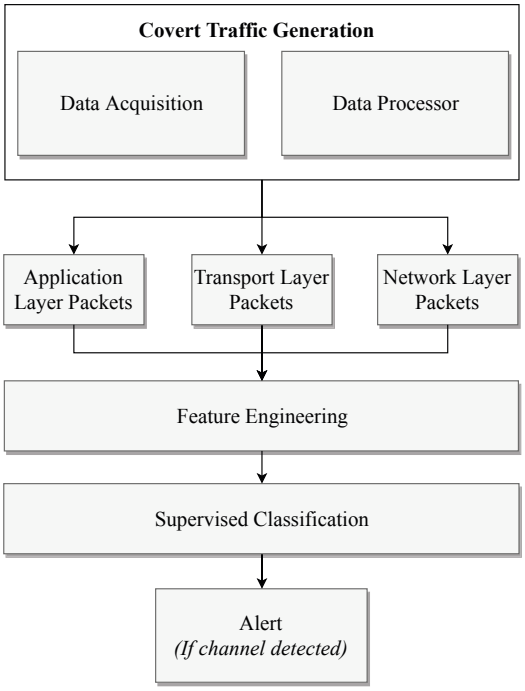

Fig. 1. Framework for Network Covert Channel Detection

A. Covert Traffic Generation

The covert network channel dataset used in this study has been generated for the TCP/IP protocol with a C program [14] and for the DNS protocol with the DNS2TCP application [6]. The Wireshark application [5] was used to capture the network packets for each protocol. Covert traffic data was generated for the network layer as embedded data in the IPv4 identification field, the transport layer as embedded covert information in the TCP sequence number field, and the application layer as embedded information in the DNS domain name query and response elements. Feature selection and feature engineering techniques were applied to process and prepare the dataset for the proposed generic approach.

In our research, the most effective method for the generation of the IP and TCP network storage covert channel datasets was found to be an application developed by Craig H. Rowland [14]. We modified this program to embed a network covert channel in the TCP and IP header fields. The program was compiled in a Linux environment and used along with Wireshark to generate and capture both normal and covert traffic for pattern recognition purposes. The covert channel technique employed in this dataset involves passing an input stream of text with one character per packet encoded in a portion of the IP ID or TCP Sequence Number fields. The final version of this dataset consists of normal and network covert channel packets in the TCP/IP protocol, which have been captured and processed into a numeric data frame.

The DNS dataset has been generated using the DNS2TCP application with Wireshark for packet capture. In order to use DNS2TCP to generate realistic data, a virtual environment

with a web server, client, and virtual network was created using Microsoft Azure [3] with Ubuntu Server 18.04 as virtual machine operating systems. DNS2TCP requires a publicly available domain name to be used for IPv4 web traffic. To accomplish this, we registered a public domain name through the Namecheap [4] domain registrar service. For creating a DNS tunnel outside the internal network, the tool also required the use of a DNS name server with 2 sub domains to direct the DNS traffic through. The FreeDNS DNS name service [1] was used to implement these sub domains. The first subdomain with a type of NS served as the subdomain address, *e.g.*, idns1.info.tm, through which the traffic is tunneled. The second subdomain was of type A with its destination set to the IP address of the DNS2TCP server. The DNS2TCP tool was installed on a client to facilitate the generation of the covert channel. This client had to be supplied with an argument of the subdomain, *e.g.*, id1.info.tm, specified in the server application. This establishes the covert channel between the client and the server as a tunnel through the DNS protocol. The covert channel communication was then encoded in the unused space of the query and response domain name fields of the DNS protocol.

Wireshark was again used to capture the DNS packets of several hundred DNS2TCP tunneled website visits as well as normal DNS queries. Next, the covert traffic packets were combined with normal DNS traffic. These packets were then processed with feature engineering methods using Python as described in the following section.

B. Feature Engineering

The captured packets' complexity as well as differences in packet fields for each protocol make an additional processing step necessary to prepare the data for the machine learning model. After generating the covert network traffic, the first step was to develop the data processor engine that works with various OSI layers, excluding the physical layer, by differentiating between protocol specific header elements. In our study, we used the IPv4 protocol for the network layer, the TCP protocol for the transport layer, and the DNS protocol for the application layer. The data processing engine accepts a set of captured network packets and returns a dataset with a generic template that is ready to be analyzed for feature selection. The feature engineering engine is necessary for processing network traffic packet information to fit into a classifier for further training and detection. This process involves feature selection of necessary fields in each protocol as well as transformation of the packets into the correct format to be read into the model.

After determining the protocol, the necessary features are extracted from the packet stream. The feature selection uses a correlation coefficient and domain knowledge to select features in the dataset [9]. For example, it eliminates TCP/IP packet elements that are not utilized or suitable for covert communication, such as the IP Differentiated Services fields and TCP window size field. The module then returns the selected features to the engine that focuses on processing the potentially compromised features, along with the header elements determined to be necessary for detection. Next, the engine performs cross-validation. The resulting dataset containing both normal and covert channel traffic is ready to run for experimentation with any machine learning technique.

C. Supervised Classification

The processed and labeled network dataset is later fed into the supervised classification machine learning model to detect traffic containing network covert channels via analysis and classification of the data. We experimented with various supervised classification techniques, *e.g.*, Logistic Regression, Support Vector Machines (Linear Kernel and Gaussian Kernel), K-nearest Neighbor, and Decision Tree. As covert channels are identified, the malicious packets are logged accordingly.

In our protocol independent approach (as illustrated in Fig. 1), the data processor engine was developed in Python 3.7. The machine learning models were implemented using scikit-learn, an open source data mining and data analysis tool [13].

IV. Results

The following sections discuss the generated covert channel traffic at each of the network, transport, and application layers as well as detection results.

A. Covert Channel in Network Layer

The IP dataset is composed of normal and covert channel traffic. The final version of this dataset contains 113,474 observations. This covert channel was hidden in the header portion of the IPv4 packet by encoding one character per packet into the unused 16 bits of the IP Identification field of unfragmented packets as marked in Fig. 2.

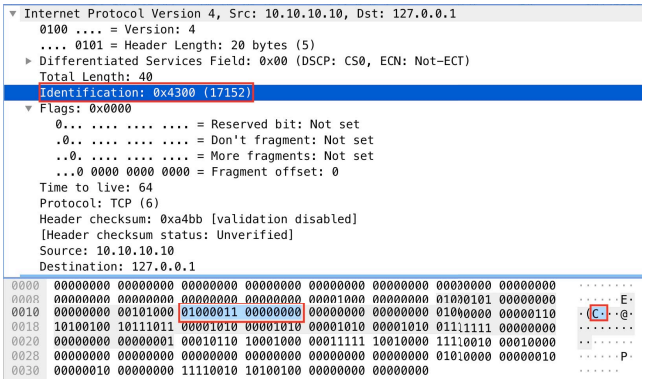

Fig. 2. Encoding random covert information in IPv4 Datagram

This dataset was used for training and experimentation with the following machine learning techniques. Fig. 3 and Fig. 4 depict the evaluation results.

1) Logistic Regression: Analysis of network covert channel detection for the network layer was first performed with a logistic regression classifier. This technique was selected to provide baseline results to compare against the rest of the techniques used. The logistic regression classifier attained an accuracy of 81.93% with a low false positive rate. To expand on those results, the method also achieved an average precision

of 83% with a statistical result of 81.12% F1 score, 94.97% precision score, and 70.79% recall score.

2) Support Vector Machine (SVM): Two kernel SVM techniques were used in our experiments: linear kernel and gaussian kernel. The linear kernel method was found more effective for the detection of covert traffic than the logistic regression classifier with an accuracy of 87.51%. This method also achieved a low false alarm rate and lower missing rate (less occurrences of true negatives). The average precision value for linear kernel was 88% while F1, precision, and recall scores were 87.68%, 95.47%, and 81.07% respectively.

The SVM using a gaussian kernel achieved great results, having an accuracy of 99.78%. This technique was found to be the most accurate of those tested for the layer, as well as achieving the highest specificity and sensitivity. The gaussian kernel classifier achieved an average precision value of 100%, and its F1, Precision, and Recall score were 99.8%, 99.66%, and 99.65% respectively.

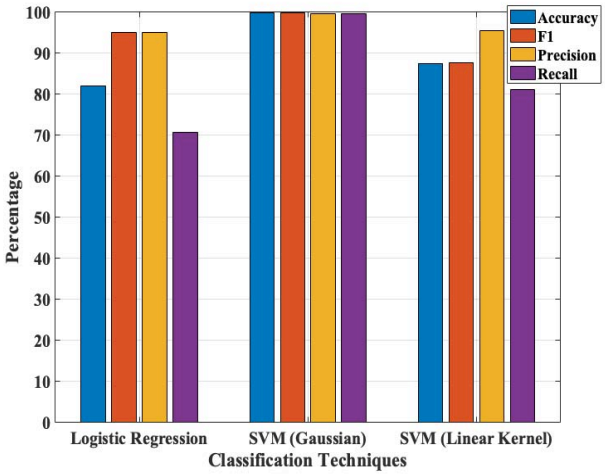

Fig. 3. Comparison of different supervised classification techniques in the Network Layer in terms of Accuracy, F1, Precision, and Recall score

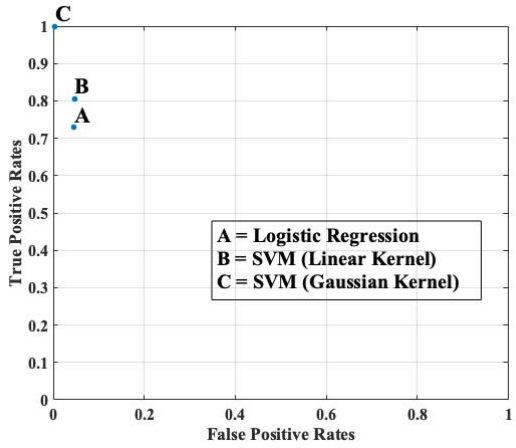

Fig. 4. ROC Space for different supervised classification techniques in the Network Layer

B. Covert Channel in Transport Layer

The TCP covert channel was implemented via embedding random covert information into the TCP sequence number field as highlighted in Fig. 5. Similar to the IPv4 dataset, covert channel and benign traffic was captured for this dataset. The final dataset contained 112,614 observations. Training and experimentation with this dataset involved the following techniques. Fig. 6 and Fig. 7 illustrated the experimental findings.

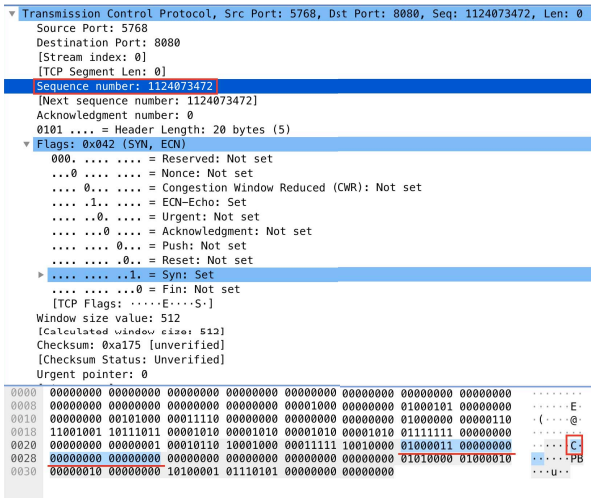

Fig. 5. Embedding random covert information in TCP Segment

1) Logistic Regression: The Logistic Regression technique was utilized to generate baseline results. This experiment resulted in an accuracy of 64.29%. The classifier achieved an average precision value of 64% while F1, precision, and recall scores are 66.68%, 68.86%, and 64.64% respectively. These results show that this technique would not be suitable for detection in the network layer.

2) Support Vector Machine (SVM): Similar to the Network Layer, SVM with both Linear Kernel and Gaussian Kernel techniques have been used for detection. The accuracy of linear kernel was 72.29% with zero false negatives. This model can be used in order to ensure minimum false alarms but is not appropriate for achieving a high detection rate. The SVM with Linear Kernel classifier obtained an average precision value of 78%. In addition, the statistical measurement of this model was 66.53%, 100%, and 49.85% for the F1, precision, and recall score.

Gaussian kernel was extremely effective in detecting TCP covert channels due to its non-linear nature. The model produced an accuracy of 99.15% with marginally more false negatives than linear kernel. The model was found to be more sensitive and specific than previously discussed techniques. Along with this high accuracy, the classifier achieved an average-precision value of 99% while F1, precision, and recall scores were 99.23%, 99.88%, and 98.59% respectively. Due to these excellent results, this classifier was found to be the most effective detection technique for the network layer.

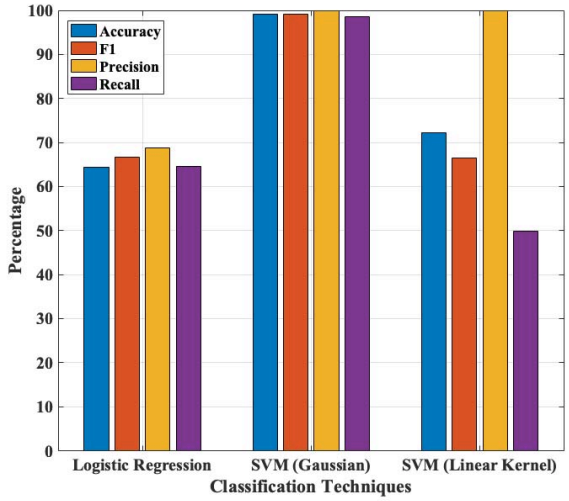

Fig. 6. Comparison of different supervised classification techniques in the Transport Layer in terms of Accuracy, F1, Precision, and Recall score

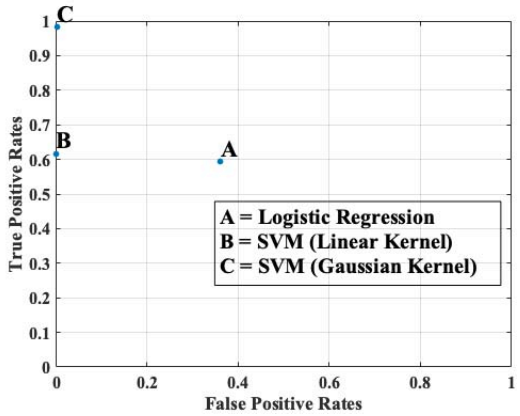

Fig. 7. ROC Space for different supervised classification techniques in the Transport Layer

C. Covert Channel in Application Layer

The Covert channel in the application layer was implemented using DNS protocol communication between the client and server. The covert communication was encoded both in the DNS client query and DNS server response name elements (such fields are marked in Fig. 8).

The generated dataset is composed of both normal DNS traffic and covert traffic. It was used in experimentation with the following classification machine learning techniques. Fig. 9 and Fig. 10 portray the experimental findings.

1) Logistic Regression: Logistic regression model gave an accuracy of 93.22% with zero true negatives. The classifier also attained an average precision value of 88% with F1, precision and recall scores of 93.58%, 87.93% and 100% respectively. These are better results than the logistic regression technique achieved for the transport or network layer but not good enough to be considered suitable for a stand-alone detection technique.

```
▸ User Datagram Protocol, Src Port: 53, Dst Port: 42899
▾ Domain Name System (response)
    Transaction ID: 0x410e
  ▸ Flags: 0x8580 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ IWUV2BXqBA.iohost.theworldofnet.online: type TXT, class IN
        Name: IWUV2BXqBA.iohost.theworldofnet.online
        [Name Length: 38]
        [Label Count: 4]
        Type: TXT (Text strings) (16)
        Class: IN (0x0001)
  ▾ Answers
    ▾ IWUV2BXqBA.iohost.theworldofnet.online: type TXT, class IN
        Name: IWUV2BXqBA.iohost.theworldofnet.online
        Type: TXT (Text strings) (16)
        Class: IN (0x0001)
        Time to live: 3
        Data length: 13
        TXT Length: 11
        TXT: AIWUAABXqEA
        TXT Length: 0
        TXT:
    [Request In: 4]
    [Time: 0.500719255 seconds]
```

Fig. 8. Encoding random covert information in DNS Query and Response

2) K-Nearest Neighbors (K-NN): The K-nearest Neighbors model was implemented by specifying the K value as 5 and was more accurate than logistic regression. The accuracy was 94.74% while the recall rate was flawless. The average precision value of this classifier was 91% while other statistical measurements were 95.26% for the F1 score, 90.95% for the precision score, and 100% for the recall score.

3) Decision Tree: Decision tree classifier was found to be very effective for detecting the existence of network covert channels in DNS packets. With an accuracy of 94.96%, the model achieved a smaller false alarm rate than the K-NN method. The statistical measurements were improved over the other techniques as well, having 91% as the average precision value, 95.45% F1 score, 91.29% precision score, and 100% recall score. These results proved the effectiveness of Decision Trees for covert channel detection in the DNS protocol.

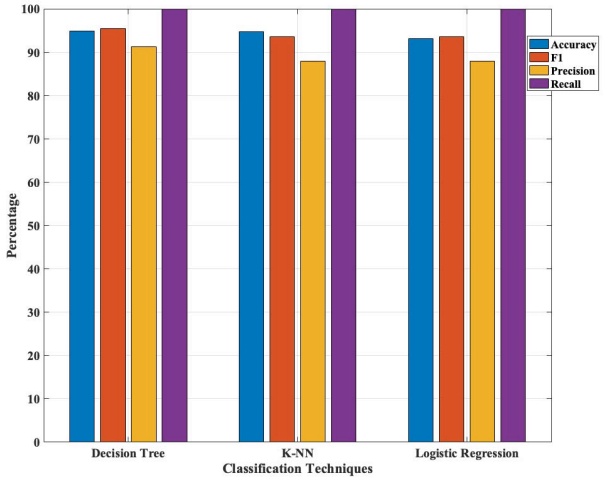

Fig. 9. Comparison of different supervised classification techniques in the Application Layer in terms of Accuracy, F1, Precision, and Recall score

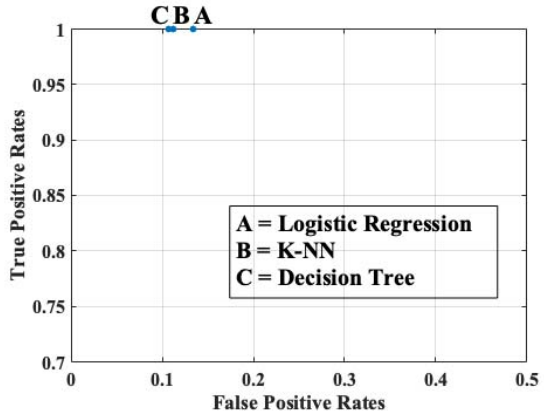

Fig. 10. ROC Space for different supervised classification techniques in the Application Layer

D. Summary

Based on our experimental results and analysis, we have determined a suite of effective techniques for protocol independent detection of network covert storage channels in the network, transport, and application layers. In both the transport and network layer, the SVM with Gaussian Kernel classifier outperformed other supervised classification techniques. However, in our application layer experiments the Decision Tree technique was found very effective.

V. Discussion

SVM with a gaussian kernel was found to be very effective for the detection of covert channels in the TCP/IP protocol suite; however, it was not suitable for the DNS protocol due to the presence of complex and distributed features. This led to experimentation with other techniques for the detection of covert channels within the DNS protocol, resulting in the decision tree classifier being found the most effective technique among those experimented with. On a side note, the DNS dataset required more effort to produce compared to the other datasets due to the need to set up a virtual server and network environment to capture packets.

Network covert channels also exist in the data link layer, which was not examined in our study. Thus, a possibility for future research would be experimentation with embedded covert communication in link frames. Other future work that could be considered to expand upon these findings is experimentation with Neural Networks. In addition, investigating the approach with covert channel data stored in the payload of the packet, and timing channels would help expand the effectiveness of the generic approach described in this study.

VI. Conclusion

This paper has described a generic approach for the detection of network storage covert channels. This was accomplished by data processing, feature engineering and application of supervised machine learning classification techniques to detect network storage covert channels in a protocol independent manner. In our experiments, we found that support vector machine (SVM) with a gaussian kernel was a very effective technique for covert channel detection in both the network and transport layers while the decision tree classifier achieved high results in the application layer. We also described our approach used in the generation of the IP, TCP, and DNS covert datasets. The dataset provides the community with a large network covert channel dataset capturing covert traffic at various network layers for future research needs and is available for access by the research community (upon request).

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments and feedback. We would also like to extend our gratitude to Dr. Edward Zieglar, National Security Agency (NSA) to collaborate with us via INSuRE Project [2]. The work reported in this paper has been fully supported by the Cybersecurity Education, Research and Outreach Center (CEROC) at Tennessee Tech University, Cookeville, TN, USA.

References

[1] Free dns hosting, dynamic dns hosting, static dns hosting, subdomain and domain hosting. Accessed on January 15, 2019.
[2] Insurehub. Accessed on March 2, 2019. URL: https://app.insurehub.org/.
[3] Microsoft azure cloud computing platform and services. Accessed on February 18, 2019. URL: https://azure.microsoft.com/en-us/.
[4] Namecheap. Accessed on December 30, 2018. URL: https://www.namecheap.com.
[5] Wireshark. Accessed on March 14, 2019. URL: www.wireshark.org.
[6] Michael Bittan, Associ, Michal, and Romain Hennion. Cyber academy : formations et certifications en cyberscurit, Jan 2019. URL: http://www.hsc.fr/ressources/outils/dns2tcp/.
[7] Greg Farnham and A Atlasis. Detecting dns tunneling. *SANS Institute InfoSec Reading Room*, 9:1–32, 2013.
[8] Butler W Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.
[9] Joseph Lee Rodgers and W Alan Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.
[10] David Llamas, C Allison, and A Miller. Covert channels in internet protocols: A survey. In *Proceedings of the 6th Annual Postgraduate Symposium about the Convergence of Telecommunications, Networking and Broadcasting, PGNET*, volume 2005, 2005.
[11] Norka B Lucena, Grzegorz Lewandowski, and Steve J Chapin. Covert channels in ipv6. In *International Workshop on Privacy Enhancing Technologies*, pages 147–166. Springer, 2005.
[12] Steven J Murdoch. Covert channel vulnerabilities in anonymity systems. Technical report, University of Cambridge, Computer Laboratory, 2007.
[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
[14] Craig H Rowland. Covert channels in the tcp/ip protocol suite. *First Monday*, 2(5), 1997.
[15] Taeshik Sohn, JungTaek Seo, and Jongsub Moon. A study on the covert channel detection of tcp/ip header using support vector machine. In *International Conference on Information and Communications Security*, pages 313–324. Springer, 2003.
[16] Eugene Tumoian and Maxim Anikeev. Detecting nushu covert channels using neural networks. *Taganrog State University of Radio Engineering*, 2005.
[17] Eugene Tumoian and Maxim Anikeev. Network based detection of passive covert channels in tcp/ip. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pages 802–809. IEEE, 2005.
[18] Steffen Wendzel, Sebastian Zander, Bernhard Fechner, and Christian Herdin. Pattern-based survey and categorization of network covert channel techniques. *ACM Computing Surveys (CSUR)*, 47(3):50, 2015.
[19] Sebastian Zander, Grenville Armitage, and Philip Branch. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys & Tutorials*, 9(3):44–57, 2007.